Assignment-11.1 – Case Study

Case Study Description

Step#1

Use the CUSTOMER and TRANSACTIONS and the data set in custs.txt and txns.txt and find out the number of transaction done by each customer

Step#2
Create a new table called TRANSACTIONS_COUNT. This table should have
3 fields - custid, fname and count.

Step#3
Write a hive query to populate TRANSACTIONS_COUNT table with the data obtained in step1

Step#4
Now lets make the TRANSACTIONS_COUNT table Hbase complaint. In the
sense, use Ser Des And Storate handler features of hive to change the
TRANSACTIONS_COUNT table to be able to create a TRANSACTIONS table
in Hbase.

Step#5
Now insert the data in TRANSACTIONS_COUNT table using the query in step
3 again, this should populate the Hbase TRANSACTIONS table automatically
(This has to be done in module 10)

Step#6
Now from the Hbase level, write the Hbase java API code to access and scan
the TRANSACTIONS table data from java level.

#SOLUTION

#Launch Hive and Create CUSTOMER table and load the data from custs.txt

```
[acadgild.mmisra ~]$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-
bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-
2.6.5/share/hadoop/common/lib/slf4j-log4j12-
1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in
jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/hive-common-
2.3.2.jar!/hive-log4j2.properties Async: true
```

```
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions.
Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X
releases.
Time taken: 0.117 seconds, Fetched: 3 row(s)
hive> use txn;
OK
Time taken: 0.018 seconds
hive> show tables;
OK
Time taken: 0.043 seconds
hive> CREATE TABLE CUSTOMER(
    > custid INT,
    > fname STRING,
    > lname STRING,
    > age INT,
    > profession STRING
    > )
    > row format delimited
    > fields terminated by ',';
OK
Time taken: 0.108 seconds
hive> LOAD DATA LOCAL INPATH '/home/acadgild/custs.txt' into table CUSTOMER;
Loading data to table txn.customer
OK
Time taken: 1.389 seconds
```

#Create table TXNRECORDS and load it with data from txns.txt

```
hive> CREATE TABLE TXNRECORDS (txnno INT, txndate STRING, custno INT, amount DOUBLE,
    > category STRING, product STRING, city STRING, state STRING, spendby STRING)
    > row format delimited
    > fields terminated by ',';
OK
Time taken: 0.08 seconds
hive>
    > LOAD DATA LOCAL INPATH '/home/acadgild/txns.txt' into table TXNRECORDS;
Loading data to table txn.txnrecords
OK
Time taken: 0.643 seconds
```

#To find the number of transactions for each customer we join both the tables based on customer ID and select only customer name and transactions in query. We group the data based on the customer ID and count the number of transactions. The query is

```
select a.custid, a.fname, COUNT(b.txnno) from CUSTOMER a join TXNRECORDS b on
    > a.custid=b.custno GROUP BY a.custid,a.fname;


hive> select a.custid, a.fname, COUNT(b.txnno) from CUSTOMER a join TXNRECORDS b on
    > a.custid=b.custno GROUP BY a.custid,a.fname;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future
versions. Consider using a different execution engine (i.e. spark, tez) or using Hive
1.X releases.
Query ID = acadgild_20180807093550_fc596616-7d66-4917-9daf-b9ca2bb79158
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-
bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-
2.6.5/share/hadoop/common/lib/slf4j-log4j12-
1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
```

```
2018-08-07 09:35:59     Starting to launch local task to process map join;
maximum memory = 477626368
2018-08-07 09:36:01     Dump the side-table for tag: 0 with group count: 10 into file:
file:/tmp/acadgild/9fa0f2d8-a54f-4b48-bd35-b0dd7f68c3e5/hive_2018-08-07_09-35-
50_264_1448860733524122480-1/-local-10005/HashTable-Stage-2/MapJoin-mapfile00--
.hashtable
2018-08-07 09:36:01     Uploaded 1 File to: file:/tmp/acadgild/9fa0f2d8-a54f-4b48-
bd35-b0dd7f68c3e5/hive_2018-08-07_09-35-50_264_1448860733524122480-1/-local-
10005/HashTable-Stage-2/MapJoin-mapfile00--.hashtable (556 bytes)
2018-08-07 09:36:01     End of local task; Time Taken: 1.965 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1533613164216_0002, Tracking URL =
http://localhost:8088/proxy/application_1533613164216_0002/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1533613164216_0002
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-08-07 09:36:10,081 Stage-2 map = 0%,  reduce = 0%
2018-08-07 09:36:16,762 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 3.07 sec
2018-08-07 09:36:24,438 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 5.72 sec
MapReduce Total cumulative CPU time: 5 seconds 720 msec
Ended Job = job_1533613164216_0002
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 5.72 sec   HDFS Read: 18260 HDFS
Write: 381 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 720 msec
```

#We can see 10 records with customer ID, name and number of transactions

```
OK
4000001 Kristina        8
4000002 Paige   6
4000003 Sherri  3
4000004 Gretchen        5
4000005 Karen   5
4000006 Patrick 5
4000007 Elsie   6
4000008 Hazel   10
4000009 Malcolm 6
4000010 Dolores 6
Time taken: 35.305 seconds, Fetched: 10 row(s)
```

#At this point of time we create a table in HBase which will hold this data. We create a table called 'TRANSACTIONS_COUNT with 'txn' column family name to hold customer name and number of transaction columns

```
Last login: Tue Aug  7 09:08:49 2018 from 192.168.56.1
[acadgild.mmisra ~]$ hbase shell
2018-08-07 09:36:43,684 WARN  [main] util.NativeCodeLoader: Unable to load native-
hadoop library for your platform... using builtin-java classes where applicable
SLF4J: Class path contains multiple SLF4J bindings.
```

```
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-
log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-
2.6.5/share/hadoop/common/lib/slf4j-log4j12-
1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

hbase(main):001:0> create 'TRANSACTIONS_COUNT','txn'
0 row(s) in 1.5120 seconds
```

#Now in Hive we create an external table called TRANSACTIONS_COUNT which uses org.apache.hadoop.hive.hbase.HBaseStorageHandler class for storage and HBASE SERDEPROPERTIES to provide mapping of hive table columns to hbase table column and column family
We also provide name of the hbase table using TBLPROPERTIES

```
hive> CREATE EXTERNAL TABLE TRANSACTIONS_COUNT(custid INT, fname STRING,count INT)
    > STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,
    > txn:name,txn:count")
    > TBLPROPERTIES("hbase.table.name" = "TRANSACTIONS_COUNT");
OK
Time taken: 1.083 seconds
```

#Then we populate TRANSACTIONS_COUNT table using the same query that we used before. We use ‘insert overwrite’ to load the table from the query

```
hive> insert overwrite table TRANSACTIONS_COUNT
    > select a.custid, a.fname, COUNT(b.txnno) from CUSTOMER a join TXNRECORDS b on
    > a.custid=b.custno GROUP BY a.custid,a.fname;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future
versions. Consider using a different execution engine (i.e. spark, tez) or using Hive
1.X releases.
Query ID = acadgild_20180807093708_fd650d69-8cae-46cd-a618-c836202b7a3d
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-
bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-
2.6.5/share/hadoop/common/lib/slf4j-log4j12-
1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
2018-08-07 09:37:15     Starting to launch local task to process map join;
maximum memory = 477626368
2018-08-07 09:37:17     Dump the side-table for tag: 0 with group count: 10 into file:
file:/tmp/acadgild/9fa0f2d8-a54f-4b48-bd35-b0dd7f68c3e5/hive_2018-08-07_09-37-
08_685_4512656815889986251-1/-local-10002/HashTable-Stage-4/MapJoin-mapfile10--
.hashtable
2018-08-07 09:37:17     Uploaded 1 File to: file:/tmp/acadgild/9fa0f2d8-a54f-4b48-
bd35-b0dd7f68c3e5/hive_2018-08-07_09-37-08_685_4512656815889986251-1/-local-
10002/HashTable-Stage-4/MapJoin-mapfile10--.hashtable (556 bytes)
2018-08-07 09:37:17     End of local task; Time Taken: 2.005 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1533613164216_0003, Tracking URL =
http://localhost:8088/proxy/application_1533613164216_0003/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill
job_1533613164216_0003
Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 1
2018-08-07 09:37:27,146 Stage-4 map = 0%,  reduce = 0%
2018-08-07 09:37:33,843 Stage-4 map = 100%,  reduce = 0%, Cumulative CPU 3.64 sec
2018-08-07 09:37:42,381 Stage-4 map = 100%,  reduce = 100%, Cumulative CPU 8.8 sec
MapReduce Total cumulative CPU time: 8 seconds 800 msec
Ended Job = job_1533613164216_0003
MapReduce Jobs Launched:
Stage-Stage-4: Map: 1  Reduce: 1   Cumulative CPU: 8.8 sec   HDFS Read: 25542 HDFS
Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 800 msec
OK
Time taken: 34.78 seconds
```

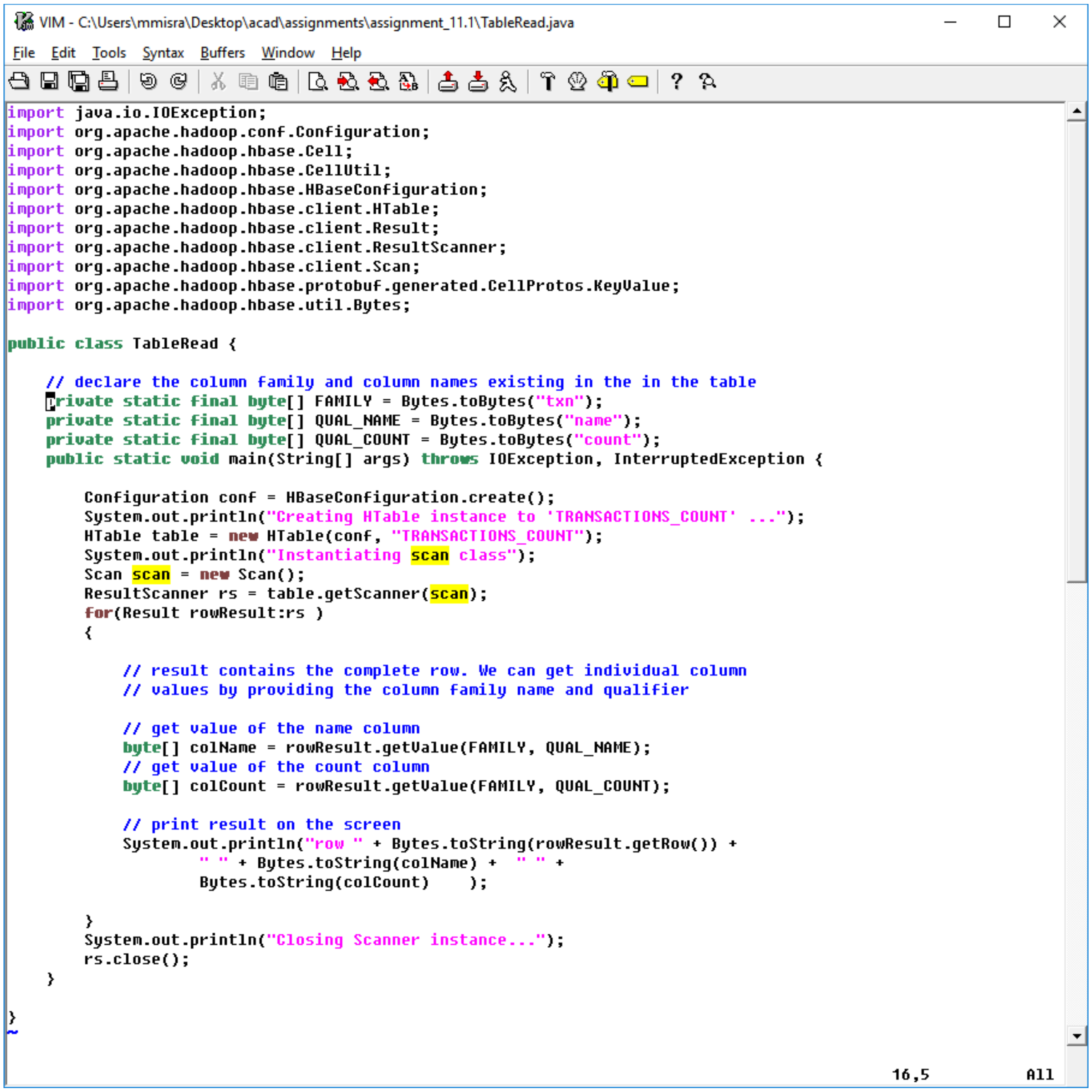
#We dump TRANSACTIONS_COUNT table to see that it has 10 records as we have seen in the previous query

```
hive> select * from TRANSACTIONS_COUNT
    > ;
OK
4000001 Kristina        8
4000002 Paige   6
4000003 Sherri  3
4000004 Gretchen        5
4000005 Karen   5
4000006 Patrick 5
4000007 Elsie   6
4000008 Hazel   10
4000009 Malcolm 6
4000010 Dolores 6
Time taken: 0.305 seconds, Fetched: 10 row(s)
hive>
```

#We goto HBase and scan the table TRANSACTIONS_COUNT to see if it has been populated from hive. We see that we have exactly 10 rows as in the hive table

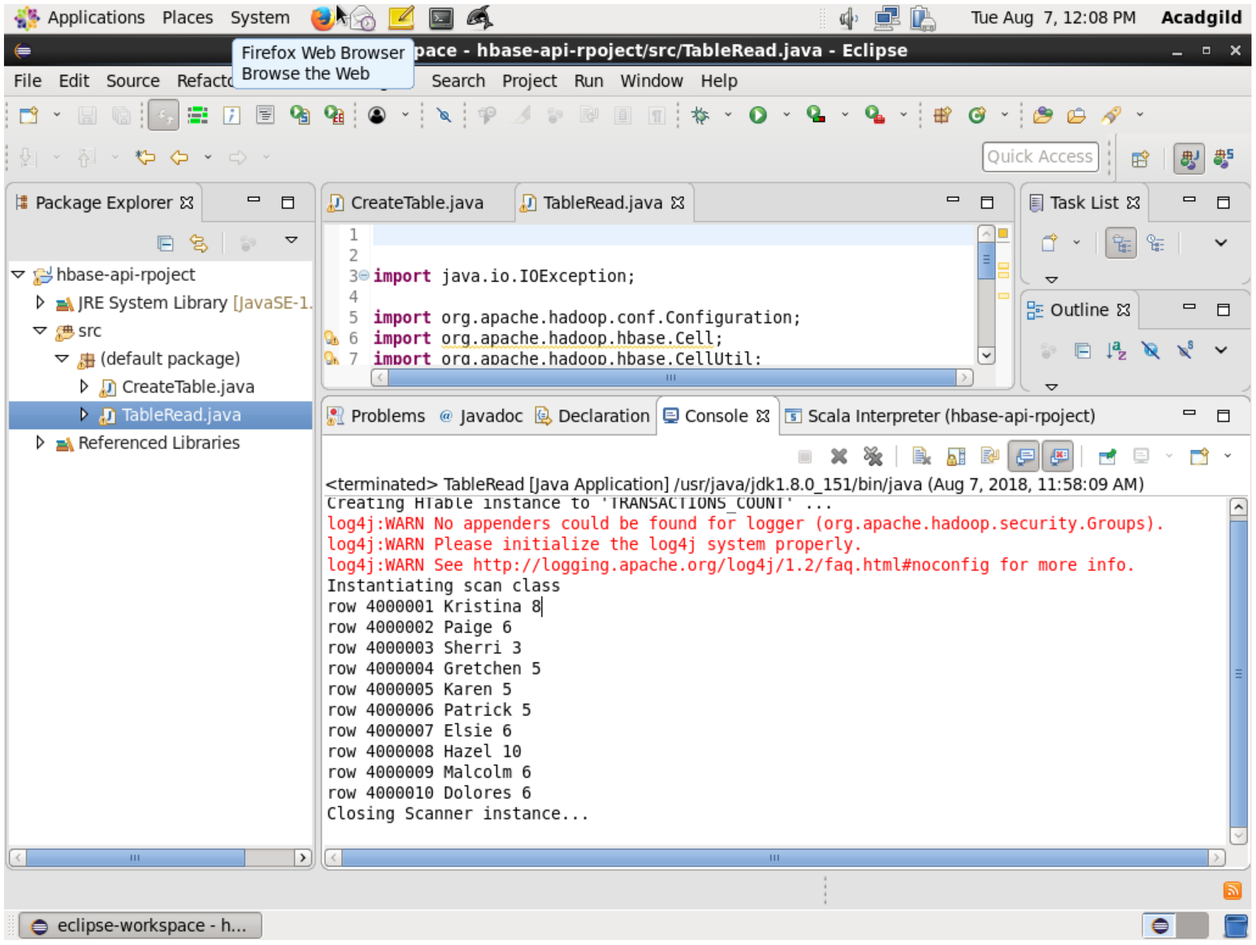
```
=> Hbase::Table - TRANSACTIONS_COUNT
hbase(main):002:0> scan 'TRANSACTIONS_COUNT'
ROW                           COLUMN+CELL
 4000001                      column=txn:count, timestamp=1533614861806, value=8
 4000001                      column=txn:name, timestamp=1533614861806,
value=Kristina
 4000002                      column=txn:count, timestamp=1533614861806, value=6
 4000002                      column=txn:name, timestamp=1533614861806, value=Paige
 4000003                      column=txn:count, timestamp=1533614861806, value=3
 4000003                      column=txn:name, timestamp=1533614861806, value=Sherri
 4000004                      column=txn:count, timestamp=1533614861806, value=5
 4000004                      column=txn:name, timestamp=1533614861806,
value=Gretchen
 4000005                      column=txn:count, timestamp=1533614861806, value=5
 4000005                      column=txn:name, timestamp=1533614861806, value=Karen
 4000006                      column=txn:count, timestamp=1533614861806, value=5
 4000006                      column=txn:name, timestamp=1533614861806, value=Patrick
 4000007                        column=txn:count, timestamp=1533614861806, value=6
 4000007                        column=txn:name, timestamp=1533614861806, value=Elsie
 4000008                        column=txn:count, timestamp=1533614861806, value=10
 4000008                        column=txn:name, timestamp=1533614861806, value=Hazel
 4000009                        column=txn:count, timestamp=1533614861806, value=6
 4000009                        column=txn:name, timestamp=1533614861806, value=Malcolm
 4000010                        column=txn:count, timestamp=1533614861806, value=6
 4000010                        column=txn:name, timestamp=1533614861806, value=Dolores
10 row(s) in 0.2060 seconds

hbase(main):003:0>
```

#We write a java program to access and print the contents of TRANSACTIONS_COUNT Table in hbase. Below is the source code for reading the table from hbase.


VIM - C:\Users\mmisra\Desktop\acad\assignments\assignment_11.1\TableRead.java
File Edit Tools Syntax Buffers Window Help

```java
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.CellUtil;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.protobuf.generated.CellProtos.KeyValue;
import org.apache.hadoop.hbase.util.Bytes;

public class TableRead {

    // declare the column family and column names existing in the in the table
    private static final byte[] FAMILY = Bytes.toBytes("txn");
    private static final byte[] QUAL_NAME = Bytes.toBytes("name");
    private static final byte[] QUAL_COUNT = Bytes.toBytes("count");
    public static void main(String[] args) throws IOException, InterruptedException {

        Configuration conf = HBaseConfiguration.create();
        System.out.println("Creating HTable instance to 'TRANSACTIONS_COUNT' ...");
        HTable table = new HTable(conf, "TRANSACTIONS_COUNT");
        System.out.println("Instantiating scan class");
        Scan scan = new Scan();
        ResultScanner rs = table.getScanner(scan);
        for(Result rowResult:rs )
        {

            // result contains the complete row. We can get individual column
            // values by providing the column family name and qualifier

            // get value of the name column
            byte[] colName = rowResult.getValue(FAMILY, QUAL_NAME);
            // get value of the count column
            byte[] colCount = rowResult.getValue(FAMILY, QUAL_COUNT);

            // print result on the screen
            System.out.println("row " + Bytes.toString(rowResult.getRow()) +
                    " " + Bytes.toString(colName) +  " " +
                    Bytes.toString(colCount)    );

        }
        System.out.println("Closing Scanner instance...");
        rs.close();
    }

}
~
                                                        16,5          All
```

# Here is the output of running this program in the VM. We can see the content is same as Hbase output of the scan command



```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.CellUtil;
```

```
<terminated> TableRead [Java Application] /usr/java/jdk1.8.0_151/bin/java (Aug 7, 2018, 11:58:09 AM)
Creating HTable instance to 'TRANSACTIONS_COUNT' ...
log4j:WARN No appenders could be found for logger (org.apache.hadoop.security.Groups).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Instantiating scan class
row 4000001 Kristina 8
row 4000002 Paige 6
row 4000003 Sherri 3
row 4000004 Gretchen 5
row 4000005 Karen 5
row 4000006 Patrick 5
row 4000007 Elsie 6
row 4000008 Hazel 10
row 4000009 Malcolm 6
row 4000010 Dolores 6
Closing Scanner instance...
```

Applications Places System
Tue Aug 7, 12:08 PM Acadgild
Firefox Web Browser
Browse the Web
eclipse-workspace - hbase-api-rpoject/src/TableRead.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
hbase-api-rpoject
JRE System Library [JavaSE-1.
src
(default package)
CreateTable.java
TableRead.java
Referenced Libraries
CreateTable.java
TableRead.java
Task List
Outline
Problems
Javadoc
Declaration
Console
Scala Interpreter (hbase-api-rpoject)
eclipse-workspace - h...